# Steps for configuring the ACS Java Filter

## Register the app in ACS

1. Register the application that wants to use ACS for authentication as a relying party in ACS.
2. Realm of the relying party should ideally be same the realm of the application.
3. Return URL of the relying party must be the URL of the application root.
4. Error URL of the relying party is optional.
5. Token format of the relying party must be either SAML 2.0 or SAML 1.1
6. Chose None as token encryption policy. The filter does not support encrypted tokens.
7. Specify the desired lifetime for the tokens.
8. Choose identity provider(s) that the application intends to use.
9. Create a new rule group or choose an existing rule group. Filter doesn’t rely on any rules. Rules are for the use of application only.
10. Select a valid certificate for signing the tokens. It can be the default namespace certificate of the ACS or a self-signed certificate of the user or a CA issued certificate for the namespace.

## Configure the filter

1. Add the jar file (filter4acs.jar) to the classpath of the application which uses it. It can either be bundled with the archive of the application or it can be copied into a directory where the jar will be included in the application classpath.
2. The filter has a dependency on Jakarta Commons Logging library version 1.1.1 or later. If commons-logging-1.1.1jar is not already in the application’s classpath, add it to the classpath.
3. Add the filter configuration to the deployment descriptor of the application, web.xml. If the application doesn’t have a deployment descriptor, create a new one.
4. Specify **com.interopbridges.acs.federation.ACSFederationAuthFilter** as the qualified name of the filter class.
5. Specify the following init-parameters to the filter
   * 1. PassiveRequestorEndpoint – It is the URL of the PassiveRequestorEndpoint for the ACS namespace where the application is registered as a relying party. This URL can be found in the FederationMetadata of the ACS namespace. Mandatory parameter.
     2. RelyingPartRealm – It is the realm of the relying party as which the application is registered in ACS. Mandatory parameter.
     3. SigningCertificateSubject – It is the DN (distinguished name) of the actual certificate which signs the tokens in ACS, not the DN of its issuers. Mandatory parameter.
     4. CertificatesPath - This can be absolute path or a path relative to environment variable. If the certificate is CA-Signed then you need to specify the paths of root and intermediate certificates as well. This parameter also supports specifying paths relative to environment variable ,

Below are some valid formats for certificate path

a)${env.JAVA\_HOME}\ssl\root.cer;${env.JAVA\_HOME}\ssl\intermediatePrimary.cer;

${env.JAVA\_HOME}\ssl\intermediateSecondary.cer

b) c:\ssl\root.cer; c:\ssl\ intermediatePrimary.cer; c:\ssl\ intermediateSecondary.cer

c) ${env.JAVA\_HOME}\selfSigned.cer

d) c:\java6\selfSigned.cer

* + 1. CacheAssertionsInSessions – It is the flag which specifies whether to use sessions to cache SAML Assertions issued by ACS on the application server or not. If its value is true, assertions will be cached in sessions. If the value is false, assertions will not be cached in sessions, but put in HTTP cookies. Optional parameter. Defaults to false.
    2. ValidateCachedAssertionSignature4EveryRequest – This parameter is considered only when CacheAssertionsInSessions parameter is set to true. It is the flag which specifies whether to validate the cached assertion for every request or not. It is useful when session data is persisted to an external storage and may be tampered. Optional parameter. Defaults to false.

1. Map the filter to the servlets or URLs which require ACS authentication.

Below is a sample web.xml

<filter>

<filter-name>ACSAuthFilter</filter-name>

<filter-class>com.interopbridges.acs.federation.ACSFederationAuthFilter</filter-class>

<init-param>

<param-name>PassiveRequestorEndpoint</param-name>

<param-value>https://iofiltersr.accesscontrol.windows.net:443/v2/wsfederation</param-value>

</init-param>

<init-param>

<param-name>RelyingPartRealm</param-name>

<param-value>http://localhost:8080/</param-value>

</init-param>

<init-param>

<param-name>CertificatesPath</param-name>

<param-value>${env.JAVA\_HOME}\ssl\root.cer;${env.JAVA\_HOME}\ssl\int1.cer;${env.JAVA\_HOME}\ssl\int2.cer</param-value>

</init-param>

<init-param>

<param-name>SigningCertificateSubject</param-name>

<param-value>CN=\*.interoperabilitybridges.com</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>ACSAuthFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

## Sample JSP Code to display claims information in stateless mode

<%@page import=*"com.interopbridges.acs.federation.ACSFederationAuthFilter"*%>

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<%@ page import=*" com.interopbridges.acs.saml.SAMLAssertion"* %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>Home Page</title>

</head>

<body>

<H1> Welcome. </H1>

This is a test page to demonstrate ACS authentication usecase using Java

Filter technology

<h4>Below are the claims that are available for the applictaion</h4>

<%

SAMLAssertion samlAssertion = (SAMLAssertion)request.getAttribute(ACSFederationAuthFilter.ATTRIBUTE\_NAME\_ASSERTION);

SAMLAssertion.Attribute[] attributes = samlAssertion.getAttributes();

**for**(SAMLAssertion.Attribute attribute : attributes) {

out.println("<b>claim name : </b>"+ attribute.getName() +"<b> value : </b>"+ attribute.getValues()[0] );

out.println("<br>");

}

%>

</body>

</html>

## Sample JSP Code to display claims information in staeful mode

<%@page import=*" com.interopbridges.acs.federation.ACSFederationAuthFilter"*%>

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<%@ page import=*" com.interopbridges.acs.saml.SAMLAssertion"* %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>Home Page</title>

</head>

<body>

<H1> Welcome. </H1>

This is a test page to demonstrate ACS authentication usecase using Java

Filter technology

<h4>Below are the claims that are available for the applictaion</h4>

<%

SAMLAssertion samlAssertion = (SAMLAssertion)session.getAttribute(ACSFederationAuthFilter.ATTRIBUTE\_NAME\_ASSERTION);

SAMLAssertion.Attribute[] attributes = samlAssertion.getAttributes();

**for**(SAMLAssertion.Attribute attribute : attributes) {

out.println("<b>claim name : </b>"+ attribute.getName() +"<b> value : </b>"+ attribute.getValues()[0] );

out.println("<br>");

}

%>

</body>

</html>